AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A method comprising:

running guest software in a processor mode that enables the guest software to operate at a privilege level intended by the guest software; and

responsive to an attempt of the guest software to perform an operation restricted by said processor mode, exiting said processor mode to transfer control over the operation to a the VMM running outside said processor mode.

2. (Original) The method of claim 1 further comprising:

responding to the operation; and

transferring control over the operation to the guest software by entering said processor mode.

3. (Original) The method of claim 2 wherein entering said processor mode includes loading processor state expected by the guest software.

4. (Original) The method of claim 1 wherein exiting said processor mode further comprises:

saving processor state used by the guest software; and

loading processor state required by the VMM.

5. (Original) The method of claim 1 wherein exiting said processor mode further comprises automatically transferring from an address space associated with the guest software to an address space associated with the VMM.

6. (Original) The method of claim 1 further comprising maintaining a flag in a processor control register to indicate whether the processor is in said processor mode.

7. (Original) The method of claim 1 further comprising reporting an ability of a processor to support said processor mode using one of a plurality of reserved feature bits that are returned in a processor register.

8. (Original) The method of claim 1 wherein exiting said processor mode comprises generating one of a plurality of interrupts and exceptions in response to the attempt of the guest software to perform the operation restricted by said processor mode.

9. (Currently Amended) The method of claim 8 wherein generating one of the plurality of interrupts and exceptions further includes:

identifying the attempt of the guest software to perform the operation restricted by said processor mode; and

determining that the attempt of the guest software would succeed if the guest software was running outside said processor mode ~~is potentially successful~~.

10. (Currently Amended) The method of claim 8 further comprising:

maintaining a redirection bitmap for the plurality of the interrupts and exceptions ~~exception~~, the redirection bitmap indicating whether each of the plurality of the interrupts and exceptions is allowed to be handled by the guest software; and

consulting the redirection bitmap to determine whether to exit said processor mode.

11. (Original) The method of claim 8 further comprising:

identifying an attempt of the guest software to modify an interrupt flag; and

modifying the interrupt flag if the interrupt flag does not control masking of interrupts.

12. (Original) The method of claim 8 further comprising:

identifying an attempt of the guest software to modify an interrupt flag; and

preventing the attempt of the guest software to modify the interrupt flag.

13. (Original) The method of claim 12 wherein preventing the attempt of the guest software to modify the interrupt flag includes providing a shadow interrupt flag for modifications by the guest software.

14. (Original) The method of claim 12 wherein preventing the attempt of the guest software to modify the interrupt flag includes generating one of the plurality of interrupts and exceptions in response to the attempt of the guest software to modify the interrupt flag.

15. (Original) A system comprising:

a memory; and

a processor, coupled to the memory, to run guest software in a processor mode that enables the guest software to operate at a privilege level intended by the guest software, to identify an attempt of the guest software to perform an operation restricted by said processor mode, and to exit said processor mode, in response to the attempt, to transfer control over the operation to a virtual-machine monitor (VMM) running outside said processor mode.

16. (Original) The system of claim 15 wherein the processor is to re-enter said processor mode after the VMM responds to the operation.

17. (Original) The system of claim 16 wherein the processor is to load processor state expected by the guest software when re-entering said processor mode.

18. (Original) The system of claim 15 wherein the processor is to save processor state used by the guest software and to load processor state required by the VMM when exiting said processor mode.

19. (Original) The system of claim 15 wherein exiting said processor mode further comprises automatically transferring from an address space associated with the guest software to an address space associated with the VMM.

20. (Original) The system of claim 15 wherein the processor is to maintain a flag in a processor control register to indicate whether the processor is in said processor mode.

21. (Currently Amended) The system of claim 15 wherein the processor is to report ~~reporting~~ an ability to support said processor mode using one of a plurality of reserved feature bits that are returned in a processor register.

22. (Original) The system of claim 15 wherein the processor is to generate one of a plurality of interrupts and exceptions in response to the attempt of the guest software to perform the operation restricted by said processor mode.

23. (Currently Amended) The system of claim 22 wherein the processor is to generate one of the plurality of interrupts and exceptions upon determining that the attempt of the guest software to perform the operation restricted by said processor mode would succeed if the guest software was running outside said processor mode ~~is potentially successful~~.

24. (Original) The system of claim 22 wherein the processor is to consult a redirection bitmap to determine whether to exit said processor mode, the redirection bitmap indicating whether each of the plurality of the interrupts and exceptions is allowed to be handled by the guest software.

25. (Original) The system of claim 22 wherein the processor is to identify an attempt of the guest software to modify an interrupt flag and to modify the interrupt flag if the interrupt flag does not control masking of interrupts.

26. (Original) The system of claim 22 wherein the processor is to identify an attempt of the guest software to modify an interrupt flag and to prevent the attempt of the guest software to modify the interrupt flag.

27. (Original) The system of claim 26 wherein the processor is to prevent the attempt of the guest software to modify the interrupt flag by providing a shadow interrupt flag for modifications by the guest software.

28. (Original) A computer readable medium that provides instructions, which when executed on a processor, cause said processor to perform operations comprising:

running guest software in a processor mode that enables the guest software to operate at a privilege level intended by the guest software; and

responsive to an attempt of the guest software to perform an operation restricted by said processor mode, exiting said processor mode to transfer control over the operation to the VMM running outside said processor mode.

29. (Original) The computer readable medium of claim 28 providing further instructions causing the processor to perform operations comprising:

responding to the operation; and

transferring control over the operation to the guest software by entering said processor mode.

30. (Currently Amended) The computer readable medium of claim 28 comprising further instructions causing the processor to perform operations comprising:

maintaining a redirection bitmap for the plurality of the interrupts and exceptions ~~exception~~, the redirection bitmap indicating whether each of the plurality of the interrupts and exceptions is allowed to be handled by the guest software; and

consulting the redirection bitmap to determine whether to exit said processor mode.